LAMP-TR-035
CAR-TR-918
CS-TR-4034

MDA 9049-6C-1250
July 1999

Building mosaics from video using MPEG motion vectors

Ryan C. Jones
Daniel DeMenthon
David S. Doermann

Language and Media Processing Laboratory
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742-3275
rjones,daniel,doermann@cfar.umd.edu

Abstract

In this paper we present a novel way of creating mosaics from an MPEG video sequence. Two original aspects of our work are that (1) we explicitly compute camera motion between frames and (2) we deduce the camera motion directly from the motion vectors encoded in the MPEG video stream. This enables us to create mosaics more simply and quickly than with other methods.

Keywords: mosaics, camera motion

The support of this research by the Department of Defense under contract MDA 9049-6C-1250 is gratefully acknowledged.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUL 1999		**00-07-1999 to 00-07-1999**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Building mosaics from video using MPEG motion vectors	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Language and Media Processing Laboratory,Institute for Advanced Computer Studies,University of Maryland,College Park,MD,20742-3275**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**29**	

LAMP-TR-035
CAR-TR-918
CS-TR-4034

MDA 9049-6C-1250
July 1999

Building mosaics from video using MPEG motion vectors

Ryan C. Jones
Daniel F. DeMenthon
David S. Doermann

Building mosaics from video using MPEG motion vectors

Ryan C. Jones
Daniel DeMenthon
David S. Doermann

Language and Media Processing Laboratory
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742-3275
rjones,daniel,doermann@cfar.umd.edu

Abstract

In this paper we present a novel way of creating mosaics from an MPEG video sequence. Two original aspects of our work are that (1) we explicitly compute camera motion between frames and (2) we deduce the camera motion directly from the motion vectors encoded in the MPEG video stream. This enables us to create mosaics more simply and quickly than with other methods.

Keywords: mosaics, camera motion

The support of this research by the Department of Defense under contract MDA 9049-6C-1250 is gratefully acknowledged.

1 Introduction

The mass digitization of video has elevated automated storage and retrieval to a grand challenge. Video sequences can store a vast amount of useful information, but redundancy between individual frames is a problem when analyzing, browsing, or searching video. Presenting the video sequence in a compact manner is a difficult challenge because eliminating redundancy could also eliminate content. The selection of static keyframes to represent a shot sequence is commonly used in commercial products, including Virage [26], Imagine Products [19] and Excalibur [6], for indexing as well as for presentation of retrieval results. Islip [10] allows users to group several keyframes into story paragraphs. These techniques are insufficient for revealing much of the content in a video sequence. Informedia [7] condenses a clip by selecting one frame from each sequence of frames in which a key word is spoken. The resulting set of frames, called Video Skims, typically represent 1/20 to 1/6 of the frames of the entire sequence. Therefore, while revealing more content than just keyframes, Video Skims are often long. In some domains it may be more appropriate to collect all the frames of each shot into one image, called a mosaic, and present the static mosaics to the user. This method overcomes the redundancy problem while revealing camera motion content more readily than other techniques. The use of mosaics for video browsing has been investigated in [23], [24], and [25]. By constructing mosaics, more efficient representations can be used in various other tasks and applications, including video analysis, editing and manipulation, surveillance, digital libraries, interactive low-bit rate video transmission, video conferencing, and high-resolution still images.

For video analysis, processing can be done more efficiently on the mosaic that represents the sequence of frames rather than processing individual frames and integrating the results. With mosaics, video edits can be applied to a whole sequence of frames more rapidly than if edits were applied to individual frames. In surveillance, each frame of a scene often covers a limited field of view, but by combining several frames of a scene into a mosaic, more can be viewed at once. In digital libraries, video data takes up a large amount of space, so mosaicking can be used to increase data compression. Instead of us-

ing reference frames for compression, mosaics can increase the predictability of individual frames, thus lowering the number of bits needed to encode individual frames. Similar techniques are used in low-bit-rate video transmission, where background information is first transmitted, and then frames predicted from the background are sent. This requires less information than if the frames are predicted from one individual frame. In video conferencing, whiteboards or blackboards can be scanned to produce better images than those that are taken with wide-angle shots.

Two forms of mosaic outputs are commonly used. One form is the static mosaic, also referred to as a "salient still" [24]. Static mosaics attempt to represent the entire sequence of frames with one image. All the frames from one sequence are aligned into one coordinate system and integrated into one image using temporal filters. *Dynamic mosaics* present frames in such a manner that each new area of a scene in a frame is added to the previous mosaic as the video runs [8]. This approach creates a sequence of mosaics which are updated after each new frame. In this work, we focus on the generation of static mosaics.

One novel feature of our technique is that rather than using computation-intensive image processing techniques to align frames of a video sequence, we estimate the camera motion by averaging motion vectors encoded in the MPEG compression scheme. The method is therefore faster than methods that rely on image processing.

2 Previous Work

To build a mosaic one must be able to align frames from a sequence and then integrate them into one image. This section will review techniques used to align frames and integrate them into an image. Typically, researchers accomplish frame alignment as follows: A model for frame-to-frame transformation is assumed; they then solve for unknowns in the model by matching points between the frames. By contrast, we compute camera motion from MPEG motion vectors and invoke the geometry of image transformations to generate a pixel mapping that will align frames with a reference frame, usually the first frame in a sequence. Morimoto and Chellappa [16] also construct mosaics using camera motion computation, but find the camera motion by tracking feature points. We

review frame alignment methods as well as work that has used MPEG motion vectors for estimation of camera motion for tasks other than mosaicking.

2.1 Frame Alignment

Techniques for frame alignment vary as to the model that is used to determine alignment between a reference frame and any other frame. Most techniques use 2D transformations (affine, projective, or quadratic) to align frames. 2D transformations are effective on static scenes such as a view of a city skyline, where most of the motion can be measured in translational parameters.

In these 2D methods the estimation model relies on features within frames. One widely used technique is hierarchical direct registration [1, 8, 9, 20–23]. This method compares the image intensities of the reference frame with those of the other frames. A Laplacian pyramid is constructed for each pair of frames using the pixel luminance intensities. For each layer in the pyramid the sum of squared differences (SSD) is computed and is used to refine the model estimates in the next layer of the pyramid.

Assume $E(u)$ represents the region which will be registered, where u is used to represent the entire motion field within that region. $\underline{X} = (x, y)$ denotes the spatial image position of a point, I the (Laplacian pyramid) image intensity and $u(x) - (u(x(x, y), v(x, y))$ the image velocity at that point. Then

$$E(u) = \sum_{X} (I(x, t) - (I(x - u(x), t - 1))^2 \tag{1}$$

$E(u)$ is minimized using a Gauss-Newton optimization technique. At each pyramid level the procedure is iterated until the error is below a threshold. This is done from top to bottom starting with a coarse representation of the frames. This method can accurately measure both small and large movements between frames. For small displacements it is not necessary to use hierarchical minimization of SSD; instead, gradient descent or Levenberg-Marquardt optimization can be used [20, 22]. These methods find locally optimal solutions.

Optical flow has also been used as a basis to compute frame transformation parameters [12, 13, 24]. With salient video stills [24], the authors estimate the affine transformation

needed to align a pair of frames of a video sequence. They model pan and tilt by gradient descent applied to the optical flow. The affine parameters are computed using a Gaussian pyramid constructed from a pair of frames. This method is effective on static scenes. In Video Orbits [13], "projective flow" is computed from the optical flow. Optical flow can be expanded to solve for as many as eight parameters, six for affine transformation, and two used for *chirping*, to model projective effects. The error between the model and optical flow is minimized. This method is effective on static scenes.

In [5], a 2D method based on features segments each frame into a foreground and background. Instead of using the whole frame of pixel intensities to perform SSD minimization to estimate the motion parameters, only the background region is processed. Dufaux and Moscheni [5] use the Gaussian pyramid of two background regions to estimate global motion as a model for camera motion. They use a phased approach. First the translational components are calculated, then affine components, and finally perspective components. This phased approach is more robust, and with a Gaussian pyramid the local minima problem is avoided because the search is non-exhaustive.

2.2 MPEG camera motion analysis

Since we compute camera motion from MPEG motion vectors, we review some work by researchers who also use these motion vectors, even if they do not apply them to mosaics. The main benefit of such techniques is decreased processing time per video sequence since the video is not decoded before processing. Most analyze the motion vectors within the MPEG video stream to compute the global motion of a shot [14, 15, 17]. In [15], motion vectors are passed through median filters to eliminate vectors that could be considered noise. Next, stationary frames are discarded, since no motion is present in such a frame. The remaining frames are grouped into *regular* (camera motion over a static background) and *irregular* groups, the latter representing global motion that doesn't accurately estimate the independently moving objects. To estimate the global motion, a least square approach is used with an affine model to fit the actual motion vectors. Regular frames characterize the motion within a shot. A bucketing technique can be used for post-production where a low-bit-rate file references the high-bit-rate

version.

In [17], a technique for indexing video based on content and camera motion is developed. First the video is segmented into shots by analyzing the I frames of an MPEG sequence. Changes in color histograms determine shot boundaries. To index the video, each segment is described by camera movement. Motion vectors from the P and B frames of the MPEG stream are extracted, and the vectors for a frame are categorized into a finite set of motion classes. Training techniques can detect stationarity, object motion, panning, tracking, zooming, and ambiguous motion between two P or B frames. The ordering of these classes for the frames in a sequence is used to categorize the motion. This is a computationally inexpensive technique but it does not accurately estimate camera motion.

In [14], camera motion was used for video indexing. Global motion is estimated using a least-square technique to match estimated affine parameters and actual motion vectors for the MPEG stream. Large discrepancies between the estimated and actual vectors are discarded and a second least-squares estimate is computed using the remaining vectors. This technique does not account for independently moving objects. Movement over large uniform areas is hard to gauge because of the amount of noise that can be present in an MPEG stream.

2.3 Frame Integration

After the motion parameters are estimated, the frames are aligned and integrated into one image. Overlapping pixels are either averaged or dropped entirely. The averaging technique includes (1) averaging all the intensity values (but this usually creates ghost images of objects not correctly aligned), (2) applying a temporal filter based on the temporal domain of the images or a predefined weighted temporal median or average, which can show the progression of moving objects [4, 8, 18, 24]. Methods to select one pixel of the overlapping images include (1) selecting the pixel with the most recent information [3, 5, 8], (2) selecting the pixel that has best resolution and quality [3, 8].

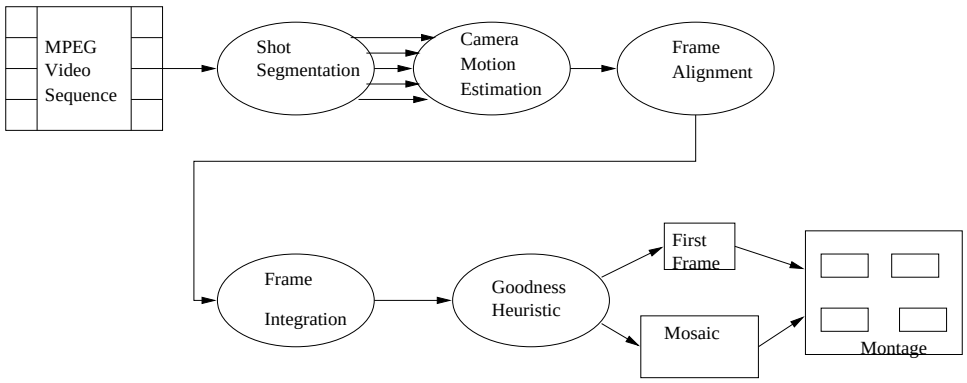

Figure 1: System diagram.

3 System Overview

A video file typically contains several scenes and/or shots. Our system builds a separate mosaic for each scene so that the user can be presented with a sequences of mosaicked images as a summary. To do this, our system requires modules for segmentation into shots, camera motion estimation, frame alignment, frame integration, and a goodness heuristic to judge the quality of the final mosaic. Scene change detection provides a file containing the indexes of the frames that begin a new scene, and is obtained by the scene change detection techniques described in [11]. The camera motion parameters are computed for each shot using the MPEG motion vectors (see Appendix), frames from a shot are aligned (Section 3.1), integrated into a static mosaic (Section 3.2) taking special care to handle zooms (Section 3.3), and a montage of all the mosaics is assembled into a single image (Section 4). This representation gives a preview of the video that can be browsed quickly. The components of the system are shown in Figure 1.

3.1 Frame Alignment

Our system aligns frames directly from camera motion estimates, bypassing any computationally intensive registration of frames. To determine the parameters needed to align different frames from a video sequence, the camera motion is computed by using the technique described in the Appendix. The technique is based on averaging the motion vectors from the macro blocks within each frame of an MPEG sequence. From the camera motion parameters the displacement between two consecutive frames in a sequence

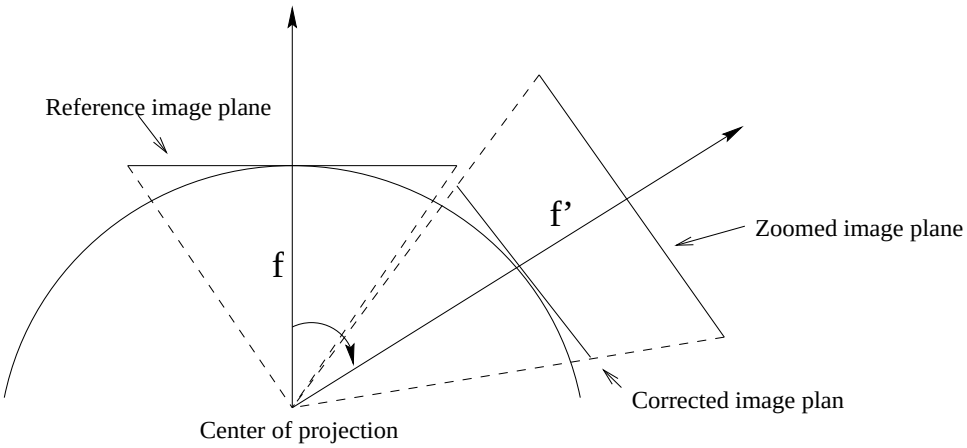

Figure 2: Cylinder diagram.

can be computed. For each successive pair of frames the pan, tilt, and zoom differences are found. To find the displacement between an initial reference frame and any other frame, the angular displacements (pan, tilt) between consecutive frames are summed up, while scale changes are performed to correct for zoom changes. The camera motion is in terms of pixel length. Thus each frames is aligned directly with the current reference frame using the constructed mosaic as the reference.

In our work the first frame of a sequence is used as the reference frame. This is a good selection if there are clean breaks between shots; otherwise, if fades are detected for shot transitions, a later frame should be used because the fade may produce a false detection of camera motion which could misalign subsequent frames in the sequence. The frame buffer in which the frames are aligned is modeled as a plane wrapped around the center of projection and no 3D translation or rotation will be computed. To simplify the alignment, the focal length of the reference frame is used for successive frames. Thus individual corrected frames are projected on a common cylindrical surface (Figure 2). The details of this computation are provided in the Appendix. The mosaic of a sequence is the visual representation of the unfolding of the frames of the sequence onto a flat plane.

3.2 Frame Integration

Once the frames have been aligned on the cylinder (Figure 2), the next step is the selection of pixels to be put into the resulting mosaic. To integrate multiple frames into a single mosaic image, the first step is to find the size of the mosaic. Each frame has associated pan, tilt, and zoom increments that align it with the previous frame. The minimal bounding box is found around the upper right hand point and lower left hand point of each frame in the sequence. Overlapping pixels can be filtered in two ways. First, the overlapping pixel that is closest to its frame center is used in the mosaic, on the basis that pixels closer to frame centers tend to be less distorted than those toward the edges. For each new frame, frame pixels that belong to the area of overlap with the mosaic and are close to the frame center are used to replace previous mosaic pixels. This is good for static scenes where the camera moves over scenery, but moving objects may be removed. To show the progression of moving objects in a shot, a second option is to average all the overlapping pixels together. The appearance that is produced is that of the object fading into the background; where it was at the begining of a clip is nearly transparent and where it was at the end of a clip is nearly opaque (Figure 6).

3.3 Zooming

In our resolution frame buffer model, frames with different focal lengths are handled by projecting them onto the same cylinder as the reference frame. If during a sequence there is a zoom-in, those frames in which zooming occurs must be scaled down so that they can be aligned with frames which have been projected onto the reference cylinder (Figure 2). Zoom-out frames are enlarged as they appear on the reference cylinder. For a zoom sequence the frames that are zoomed in are more accurate. Thus these frames are selected when constructing the final image and the pixels of the overlapped unzoomed frames are discarded, if the absolute pixel selection method is chosen.

Figure 3: Panoramic view from a control tower, constructed from a 4-second clip.

3.4 Goodness heuristic

Not every video shot produces a useful mosaic. The motion parameters are prone to error; moving objects sometimes confuse the camera motion computations (see Appendix). A heuristic is used to determine whether it is better to present the mosaic of a sequence or an individual keyframe. The heuristic value is determined by the square root of the sum of squared differences of luminance intensity values at the adjoining pixels along the boundaries between frames (Figure 4).

$$\text{HeuristicValue} = \sqrt{\sum [I(x,y) - I(x+\Delta x, y+\Delta y)]^2} \tag{2}$$

If areas of the mosaic corresponding to different frames don't match well, there will be large discontinuities at the boundaries between these areas. Above a certain threshold of discontinuity, the mosaic becomes confusing and is less useful than individual keyframes. A good threshold was determined empirically.

4 Results

To evaluate the performance of this mosaicking technique we ran it on several video clips from our in-house database that demonstrate the advantages and disadvantages of our system. The length of the video sequences ranged from 3 seconds to 2 minutes.

Figure 4: Panoramic view of a conference room sequence of 600 frames. Black lines illustrate the boundaries between areas of selected frames. In this mosaic every tenth frame was considered for mosaicking.

Figure 5: Mosaic with pixel dropping filtering applied on a 80-frame sequence.

The example in Figure 3 shows how good the results can be when there is little or no movement of objects in the video sequence. In the video sequence that produced this result there were 117 frames, all of which were considered when creating the mosaic. The camera pans from left to right. The mosaic created in Figure 4 came from a video sequence in which the camera moved left to right followed by a zoom-in. One can see the zoom by the bounding boxes on the left side. For this video sequence the computation of the zoom parameter from the MPEG motion vectors was accurate enough to produce a nearly seemless integration of video frames. The correlations of the MPEG encodings used to produce the motion vectors gave better results in some parts of the sequence than others.

While camera motion is revealed in a mosaic, whether objects moving in the shot

Figure 6: Mosaic with temporal averaging filter applied to an 80-frame sequence.

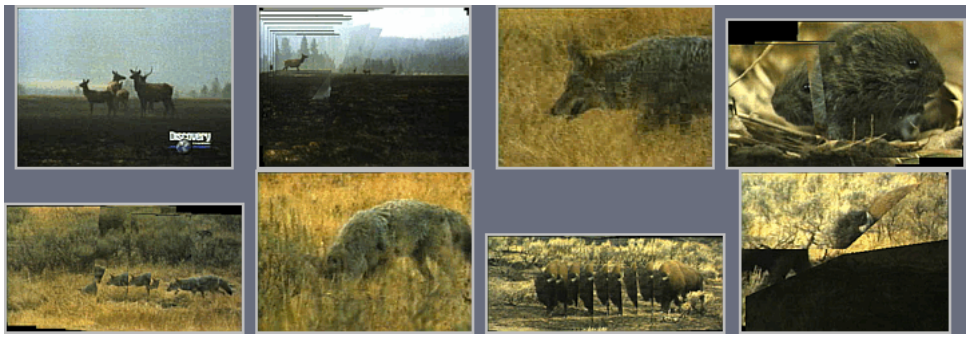

Figure 7: Montage of a 60-second video clip.

are present may not be revealed. By using an averaging technique for image integration, the progression of an object can be detected. In Figure 5 and Figure 6, a mosaic is constructed for a sequence in which the camera pans left to right while tracking a player as he is backing up from the ball. While this action is evident with a pixel averaging technique, it would not be identified if the frames were integrated by selecting one of the overlapping pixels (Figure 6).

One way of browsing a long video clip is to examine representative frames throughout a sequence. While content is revealed, camera motion is not. Our system can produce a montage of all the mosaics that reveals both content and camera motion (Figure 7). This allows users to browse the video sequence quickly. This type of representation allows users to get a better sense of the action that is present in a video sequence than by looking at representative frames. The montage clearly reveals the presence of any panning and tilting present in the video sequence. Zooming is harder to detect because all zoomed

Figure 8: Montage of a 2-minute clip.

Figure 9: 10th and 275th frames of the third mosaic in Figure 8.

frames are scaled to align with the representative frames. If the field of view does not change from the representative frames the zoom will only enhance what is present in the first frame. After careful examination a zoom sequence can be detected in the second mosaic on the first row. The deer on the right are sharper than those in the center. Video shots that did not produce good alignments were represented by the keyframes. For example in Figure 7 the third image of the first row in the montage is a keyframe. The mosaic was discarded in favor of the first frame of the sequence.

In Figure 8, an observer can detect that most of the motion is present in the first row. In this row the second mosaic contains black space. Black space will be present in a mosaic when the camera motion combines panning and tilting. The third (from left) mosaic is a pan. The height was shortened to keep the width the same as that of the

Figure 10: Montage of a 30-second clip.

keyframe. Mosaicking has problems with handling fades between shots. Without a clean shot break, the faded in/out frames will blur the final mosaic if pixels from these frames are selected during integration. The second mosaic of Figure 8 handles a case that other mosaic techniques have problems with, a forward moving camera. In this scene a camera flies over a canyon. The later frames are reduced and projected to the same image plane as the first (Figure 9). In Figure 10, the first four mosaics were not aligned well enough, so the first frame from each scene was used in the montage; while the last mosaic is a tilting sequence in which the width is shortened to keep the height the same as that of the keyframes.

5 Conclusion

By summarizing sequences of video frames into one mosaic image, more information can be viewed at once than by browsing keyframes. Long panning and tilting sequences become easier and faster to understand when viewing mosaics than when viewing the video sequentially or with keyframes. Our technique for constructing mosaics is novel in two respects. First, we explicitly compute the camera motion between frames and are able to construct a mosaic image because we know where the image planes of the camera are in space at the times when the frames are taken. The mosaic construction consists of projecting the pixels from the individual image planes onto a common cylindrical surface which is flattened to display the mosaic (Figure 2). Second, we use the motion vector

information contained in the MPEG encoding of the video to compute the camera motion between frames, including the zoom factor. By using the MPEG vectors to determine pan, tilt, and zoom, no other image processing is needed to align the frames. Therefore this method is faster at creating mosaics than methods that rely on image processing techniques. Ultimately, this method will be useful for browsing and indexing large video databases.

6 Appendix

The image flow field induced by the camera motion generally depends not only on the camera motion parameters, but also on the 3D structure of the scene. Recovering both the 3D structure and the camera motion from the flow field is an active area of research which is starting to show some success [2]. However, this research is of limited use for broadcast videos and movies. Sequences where a camera merely scans a scene of fixed objects are quickly boring and are uncommon. We would like to recover general camera motions and reconstruct the scene when humans and animals keep moving around in the field of view while the camera operator plays with the zoom of the camera, but this is clearly a difficult problem. Unfortunately, this is the most common situation in broadcast video. Most situations become ambiguous, and only common-sense knowledge can point to the correct answer. This is the situation we encounter in everyday life, where we move around in the middle of things that may or may or may not themselves move around. In some cases, our own interpretation of what is moving may be wrong. For example, we can see the moon cruising along between the tree tops beside our car at night, and may remember how puzzling this "behavior" was when we were children. In other instances, as when sitting in a train in a station while another train seem to be starting next to us, we are not sure if we are moving in a fixed scene or the other train is starting. We cannot expect an artificial vision system which does not perform any object recognition and scene interpretation to be less confused than humans are in some ambiguous situations.

Therefore, our attempt to solve the problem of recovering camera motion when moving entities are present in the scene makes use of the following simplifying assumptions:

1. When a camera (fixed or moving) views regions that seem to be moving with respect to each other, the background on which the camera motion is computed corresponds to the largest area of the field of view consistent with a camera motion.

2. The camera motion model is restricted to a combination of pan, tilt and zoom, with no translation and no swing.

The first assumption is an attempt to make a choice in the "train station" ambiguity

described above. If the majority of the field of view is moving consistently and in a way consistent with a camera motion (this motion could be zero), it is assumed to correspond to the scene's background, and the camera motion is computed using this background. Improvements would include giving more weight to the periphery of the field of view (because moving objects being tracked are generally centered in the image) and giving less weight to compact regions (because they tend to be moving objects and not background). The fact that a region is not moving is not a clue that it is the background because moving objects of interest have no image flow when they are tracked and kept at the center of the image.

The second assumption limits the degrees of freedom of the model of camera motion. One reason for doing this is to limit chances of misinterpretation in the presence of moving objects. For example, a human subject lifting up one arm and lowering the other could be interpreted as a camera swing because image flow vectors are going up on one side of the image and down on the other. Without the freedom to accept swing, the arm motions are not interpreted as due to camera motion. When two persons walk toward each other, the image flow generated on the persons could be interpreted as due to a camera translating around two still persons and progressively aligning itself with the two persons. With a model that does not allow camera translation, this case has a greater chance of being correctly interpreted.

These camera motion restrictions are not unreasonable in view of the type of data we are analyzing. In videos produced for broadcasting, camera translations are rare, partly because they are difficult to accomplish without jitter. Camera rotations are comparatively easy to produce smoothly with a shoulder camera or using a tripod. Therefore the most common camera motions are rotations. Among them, only pan and tilt rotations are common. Tripods generally do not allow rotations around the camera optical axis (swing). This motion is not comfortable to the viewer who gets the impression he is falling sideways. On the other hand, zoom actions are very common, and need to be recovered. Against the objection that zooming is not strictly speaking a camera motion, it can be argued that part of the camera is moving (distances in the lens assembly vary). In terms of image generation, zooming corresponds to variation in the focal length of the

camera, i.e. a change in the distance of the image plane with respect to the projection center.

What happens in cases where camera translation is present and we use a motion model that assumes that translation is zero? If the scene is approximately planar or has small variations in depth compared to the average scene depth, rotation and translation are difficult to disambiguate, and the image flow field can be approximately explained by a pure camera rotation instead of a translation.

6.1 Scene Points

In the time interval between two frames of a video clip, the camera may be subjected to angular motions around its center of projection O. These are called tilt (t) around an axis OY parallel to the rows of the camera sensor, and pan (p) around an axis OX parallel to the columns of the sensor. Swing (around an axis perpendicular to the sensor) is assumed to be zero, as discussed above. In addition, a zooming action may take place, obtained by a change of camera focal length from f to f', characterized by a zoom factor $\zeta = f'/f$.

A world point $\mathbf{M} = (X, Y, Z)$ moves with respect to the camera frame of reference (OX, OY, OZ) to a new position $\mathbf{M}' = (X', Y, Z')$ as this frame of reference is rotated (note that the zoom factor modifies the perspective projection, not the displacement of world points relative to the camera reference frame). Since the pan and tilt can only be small in the 1/30 second separating the capture of the two frames, their sines and cosines can be approximated by their first-order terms in the equations for the camera rotation.

The transformation of point M into point M' is expressed by a rotation matrix which can be approximated for small pan and tilt angles as

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \simeq \begin{bmatrix} 1 & 0 & -p \\ 0 & 1 & t \\ p & -t & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3}$$

We introduce an additional approximation. From Eq. 3, the relation between Z' and

Z is

$$Z' = pX - tY + Z = Z(1 + pX/Z - tY/Z) \tag{4}$$

The terms pX/Z and tY/Z are of order 2 compared to the term 1 if X/Z and Y/Z are small, since p and t are themselves small angles. This is the case if the camera has a long focal length or if we consider only scene points that project relatively close to the image center (because, as explained below, $X/Z = x/f$ and $Y/Z = y/f$, where x and y are the coordinates of the image projections of the scene points). Then this relation reduces to $Z' = Z$, and Eq. 3 can be rewritten as

$$\begin{bmatrix} X' \\ Y' \\ Z \end{bmatrix} \simeq \begin{bmatrix} 1 & 0 & -p \\ 0 & 1 & t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

6.2 Perspective Projection

The point $m = (x, y)$ is the perspective projection of M with a pinhole camera of focal length f. The point $m' = (x', y')$ is the perspective projection of M' with a new focal length f'. Therefore

$$\mathbf{M} = (X, Y, Z) = (\frac{xZ}{f}, \frac{yZ}{f}, Z)$$

$$\mathbf{M}' = (X', Y', Z) = (\frac{x'Z}{f'}, \frac{y'Z}{f'}, Z)$$

$$\begin{bmatrix} \frac{x'Z}{f'} \\ \frac{y'Z}{f'} \\ Z \end{bmatrix} \simeq \begin{bmatrix} 1 & 0 & -p \\ 0 & 1 & t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{xZ}{f} \\ \frac{yZ}{f} \\ Z \end{bmatrix} \tag{5}$$

The column vectors of the left and right hand sides are both proportional to Z. We can then divide both sides by Z. We also factor f and f', and obtain

$$\begin{bmatrix} x' \\ y' \end{bmatrix} \simeq \frac{f'}{f} \begin{bmatrix} 1 & 0 & -pf \\ 0 & 1 & tf \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{6}$$

or simply

$$\begin{aligned} x' &= \zeta(x - fp) \\ y' &= \zeta(y + ft) \end{aligned}$$

6.3 Motion Vectors

We define a vector $\mathbf{mm'}$ as the image motion vector corresponding to the motion of the world point M from M to M'. Its coordinates are $dx = x' - x$, $dy = y' - y$. The previous equations can be rewritten as

$$dx = (\zeta - 1)x - \zeta fp \quad (7)$$

$$dy = (\zeta - 1)y + \zeta ft \quad (8)$$

Note that generally the focal length f for the first image is unknown. We are able to retrieve the products fp and ft rather than p and f. These are the arcs traced by an arm of length f rotating by the pan angle p and the tilt angle t. For cylindrical mosaicking, these arcs are precisely the quantities that are needed, rather than the corresponding angles. In the following, we describe how to compute the quantities fp, ft, and ζ by averaging the field of motion vectors over the image.

6.4 Averaging over a whole frame

In an MPEG clip the motion vectors $\mathbf{m}_i\mathbf{m'}_i$ are known, as well as their positions m'_i at the centers of macroblocks. Therefore we can write equations such as Eqs. 7 and 8 *for each macroblock for which a motion vector is defined.*

With enough motion vectors (more than one), the unknowns $\zeta - 1$, (ζfp) and (ζft) can be computed, and once the zoom factor ζ is found, fp and ft can be deduced. Rather than attempting to solve a large overdetermined linear system to retrieve the unknowns, we take advantage of the fact that the coordinates x and y where the motion vectors (dx, dy) are defined are distributed on a regular grid.

First note that if we sum Eqs. 7 and 8 over the N points of the grid of macroblocks, the equations become

$$\sum_N dx_i = (\zeta - 1)\sum_N x_i - N\zeta fp$$
$$\sum_N dy_i = (\zeta - 1)\sum_N y_i + N\zeta ft$$

The terms x_i are the x coordinates of the macroblocks in the image. These coordinates are positive on the right of the image center and negative on the left of the image center, and the positive and negative terms cancel out in the sum $\sum_N x_i$. Thus

$$fp = -\frac{\sum_N dx_i}{N\zeta} = -\frac{\overline{dx}}{\zeta} \quad (9)$$
$$ft = \frac{\sum_N dy_i}{N\zeta} = \frac{\overline{dy}}{\zeta} \quad (10)$$

Above we used the fact that $\frac{1}{N}\sum_N dx_i$ is the mean of the motion vector component dx over the grid of macroblocks, denoted by $\overline{dx}$. We use this notation for all means.

We were able to write $\sum_N x_i = 0$ because the function $f(x) = x$ is an odd function for the variable x, and any odd function $f(x)$ of x would have the same property $\sum_N f(x_i = 0$ when sampled and summed over a regular grid. We now use this property to extract the unknown ζ from the equations. To eliminate the pan and tilt terms and preserve the terms containing ζ, we multiply both sides of Eq. 7 by an odd function $f(x)$, and Eq. 8 by an odd function f(y), then we sample and sum the results over the grid of N macroblocks. We obtain

$$\sum_N f(x_i)dx_i = (\zeta - 1)\sum_N x_i f(x_i) - \zeta fp\sum_N f(x_i)$$
$$\sum_N f(y_i dy_i) = (\zeta - 1)\sum_N y_i f(y_i) + \zeta ft\sum_N f(y_i)$$

Since $f(x)$ is taken to be an odd function, $\sum_N f(x_i) = 0$. We obtain

$$\overline{f(x)dx} = (\zeta - 1)\overline{xf(x)}$$
$$\overline{f(y)dy} = (\zeta - 1)\overline{yf(y)}$$

The zoom factor ζ can be computed from either of these two equations, or from the average of the two solutions:

$$\zeta = 1 + 0.5 \left(\frac{\overline{f(x)dx}}{\overline{xf(x)}} + \frac{\overline{f(y)dy}}{\overline{yf(y)}} \right) \tag{11}$$

In practice, we use $f(x) = \sin(kx)$. The factor k is selected to give little or no weight to the motion vectors at the edges of the image, which tend to be noisy. In addition, as was noted above, the approximation of Eq. 4 by $Z' = Z$ is more accurate if we discard points far from the image center.

6.5 Keeping the Grid Symmetrical with Missing Data

It is important to remember that in the above derivation sums over samplings of odd functions cancel out only when the summation is performed over a symmetric grid. In practice there are grid points where the motion vector coordinates dx_i and dy_i are not available: they have been found to belong to a moving object, or they belong to a region that has little texture. Also, motion vectors may not be known because the corresponding macroblocks are *intracoded*: the MPEG encoding process does not define a motion vector for them. If we ignored only these macroblocks when we computed sums, we would no longer be summing over a symmetric grid. To maintain summation over a symmetric grid, we must also ignore the macroblocks that are symmetric to these discarded macroblocks in the image.

Therefore, our implementation consists of first flagging the macroblocks that cannot be used. When a function that is odd with respect to x is required, a grid symmetrical with respect to the y axis must be used; thus a flag must also be set for the macroblocks that are symmetrical and have the same y coordinates but the opposite x coordinates. Averages are computed only over the macroblocks that have not been flagged. The approach is identical when the roles of x and y are swapped.

6.6 From MPEG encoding to camera motion

In MPEG encoding, for each block a motion vector represents the displacement between the block and the region that best correlates with it in another frame which may be several

frames away. This displacement is expressed in half pixels in the x and y directions. We have found that motion vector data between consecutive frames are much less noisy than others; therefore we use only subvectors when we have a choice. For an I frame or a P frame, we use the backward predicted motion vector of the previous B frame. For a B frame, we use whatever motion vector is available, and we average the forward and backward predicted motion vectors when both are available. Details are provided in [11] and [15] with slightly different implementations.

6.7 Composition of camera motions over sequences of frames

The zoom factor ζ is the ratio between the focal length f' of the present frame and the focal length f of the previous frame. Therefore, if we want to find the change of zoom over several frames, we need to compose zoom factors between pairs of frames; these should be composed by multiplication. If we find a zoom factor $\zeta_{1,2}$ between frame 1 and frame 2, and a zoom factor $\zeta_{2,3}$ between frame 2 and frame 3, the zoom factor between frame 1 and frame 3 is $\zeta_{1,3} = \zeta_{1,2}\zeta_{2,3}$.

Pan and tilt between two frames are arcs of motion with a radius equal to the focal length at the first of the two frames. If we find a pan arc ($f_1 p_{1,2}$) between frame 1 and frame 2, and a pan arc ($f_2 p_{2,3}$) between frame 2 and frame 3, we would like to cumulate pan angles, or equivalently pan arcs around a circle. We cannot just add these two arcs, because they correspond to two different radii f_1 and f_2. Instead, we have to normalize all the arcs to the same focal length, for example the general focal length f_1. We can use the zoom ratios to perform this normalization:

$$f_1 p_{2,3} = \frac{f_1}{f_2} f_2 p_{2,3} = \frac{1}{\zeta_{1,2}} f_2 p_{2,3}$$

Taking f_1, the focal length of the first frame of the shot, as the reference radius with unit length, $f_1 p_{2,3}$ is simply the angle of pan of the camera in radians.

What is the cumulated angle of pan $p_{1,4}$ between frame 1 and frame 4?

$$f_1 p_{1,4} = f_1 p_{1,2} + \frac{1}{\zeta_{1,2}}(f_2 p_{2,3}) + \frac{1}{\zeta_{1,2}\zeta_{2,3}} f_3 p_{3,4}$$

or more generally

$$f_1 p_{1,i} = \sum_i \frac{(fp)_i}{\zeta_{1,i}}$$

with $\zeta_{1,i} = \prod_i \zeta_i$. Here we have reverted back to the simpler notation, where ζ_i is the zoom factor and $(fp)_i$ is the pan arc from frame i to frame $i+1$ (with the first frame counted as frame 1).

Similarly, for the camera tilt we have

$$f_1 t_{1,i} = \sum_i \frac{(ft)_i}{\zeta_{1,i}}$$

These are the cumulated arcs in the pan and tilt of the camera motion around a sphere of radius equal to the initial camera focal length. They can be used to build a spherical mosaic of a scene. The problem with a spherical mosaic is that for large angles it cannot be shown on a sheet of paper without distortion. However, generally the tilt angle is small, and the mosaic can be approximated by a cylindrical mosaic which can easily be flattened.

References

[1] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierachical model-based motion estimation. In *Proceedings of the European Conference on Computer Vision*, pages 237–252, 1992.

[2] T. Brodsky, C. Fermuller, and Y. Aloimonos. Beyond the epipolar constraint. In *Proceedings of the Workshop on 3D Structure from Multiple Images of Large Scale Environments*, 1999.

[3] P.J. Burt, M. Hansen, and P. Anandan. Video mosaic displays. In *Proceedings of SPIE,* Volume 2736, 1996.

[4] J. Davis. Mosaic of scenes with moving objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 354–360, 1998.

[5] F. Dufaux and F. Moscheni. Background mosaicking for low bit rate video coding. In *Proceedings of the International Conference on Image Processing*, pages 673–676, 1996.

[6] Excalibur. *http://www.excalib.com.*

[7] Informedia. *http://www.informedia.cs.cmu.edu.*

[8] M. Irani, P. Anandan, and S. Hsu. Mosaic based representations of video sequences and their applications. In *Proceedings of the International Conference on Computer Vision*, pages 22–30, 1995.

[9] M. Irani, B. Russo, and S. Peleg. Computing occluding and transparent motions. *International Journal of Computer Vision*, 12:5–16, 1994.

[10] Islip. *http://www.islip.com.*

[11] V. Kobla, D.S. Doermann, K-I. Lin, and C. Faloutsos. Compressed domain video indexing techniques using DCT and motion vector information in MPEG video. In *Proceedings of the SPIE conference on Storage and Retrieval for Image and Video Databases V,* Volume 3022, pages 200–211, 1996.

[12] S. Mann and R.W. Picard. Virtual bellows: Constructing high quality stills from video. In *Proceedings of the IEEE International Conference on Image Processing*, pages 363–367, 1994.

[13] S. Mann and R.W. Picard. Video orbits of the projective group; a simple approach to featureless estimation of parameters. MIT Media Laboratory Perceptual Computing Section No. 338, 1995.

[14] J. Meng and S.-F. Chang. CVEPS: A compressed video editing and parsing system. In *Proceedings of the ACM Multimedia Conference*, pages 43–53, 1996.

[15] R. Milanese, F. Deguillaume, and A. Jacot-Descombes. Video segmentation and camera motion characterization using compressed data. In *Proceedings of the SPIE*

Conference on Multimedia Storage and Archiving Systems II, Volume 3229, pages 79–89, 1997.

[16] C. Morimoto and R. Chellappa. Fast 3D stabilization and mosaic construction. In *Proceedings of the IEEE Conference on on Computer Vision and Pattern Recognition*, pages 660–665, 1997.

[17] N.V. Patel and I.K. Sethi. Video shot detection and characterization for video databases. *Pattern Recognition*, 30, 1997.

[18] S. Peleg and J. Herman. Panoramic mosaics by manifold projection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 338–343, 1997.

[19] Imagine Products. *http://www.imagineproducts.com.*

[20] H.S. Sawhney, S. Ayer, and M. Gorkani. Model-based 2D and 3D dominant motion estimation for mosaicing and video representation. In *Proceedings of the International Conference on Computer Vision*, pages 583–590, 1995.

[21] Y. Seo, S. Choi, H. Kim, and K.S. Hong. Where are the ball and players? Soccer game analysis with color-based tracking and image mosaick. In *Proceedings of the International Conference on Image Analysis and Processing*, 1997.

[22] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphic and Applications*, 16(2):22–30, 1996.

[23] Y. Taniguchi, A. Akutsu, and Y. Tonomura. PanoramaExcerpts: Extracting and packing panoramas for video browsing. In *Proceedings of the ACM Multimedia Conference*, pages 427–436, 1997.

[24] L. Teodoiso and W. Bender. Salient video stills: Content and context preserved. In *Proceedings of the ACM Multimedia Conference*, pages 39–46, 1993.

[25] Y. Tonomura, A. Akutsu, K. Otsuji, and T. Sadakata. Videomap and VideoSpace-Icon for anatomizing video content. In *Proceedings of INTERCHI*, pages 131–138, 1993.

[26] Virage. *http://www.virage.com.*